LAMPIRAN

Lampiran 1

```
/*
  TinyGPS++ - a small GPS library for Arduino providing
universal NMEA parsing
  Based on work by and "distanceBetween" and "courseTo"
courtesy of Maarten Lamers.
  Suggestion to add satellites, courseTo(), and cardinal() by
  Matt Monson.
Location precision improvements suggested by Wayne Holder.
Copyright (C) 2008-2013 Mikal Hart  All rights reserved.
  This library is free software; you can redistribute it
  and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation;
  either
  version 2.1 of the License, or (at your option) any later
version.
  This library is distributed in the hope that it will be
  useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty
  of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
  the GNU
  Lesser General Public License for more details.
  You should have received a copy of the GNU Lesser General
  Public
  License along with this library; if not, write to the Free
  Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
  02110-1301  USA
  */

  #ifndef __TinyGPSPlus_h
  #define __TinyGPSPlus_h

  #if defined(ARDUINO) && ARDUINO >= 100
```

```
      #include "Arduino.h"
      #else
      #include "WProgram.h"
      #endif
      #include <limits.h>

      #define _GPS_VERSION "1.0.2" // software version of this
      library
      #define _GPS_MPH_PER_KNOT 1.15077945
      #define _GPS_MPS_PER_KNOT 0.51444444
      #define _GPS_KMPH_PER_KNOT 1.852
      #define _GPS_MILES_PER_METER 0.00062137112
      #define _GPS_KM_PER_METER 0.001
      #define _GPS_FEET_PER_METER 3.2808399
      #define _GPS_MAX_FIELD_SIZE 15

      struct RawDegrees
      {
        uint16_t deg;
uint32_t billionths;      bool
negative;        public:
         RawDegrees() : deg(0), billionths(0), negative(false)
         {}
      };


      struct TinyGPSLocation
      {
        friend class TinyGPSPlus;
     public:
        bool isValid() const    { return valid; }       bool
isUpdated() const  { return updated; }    uint32_t age() const
{ return valid ? millis() - lastCommitTime : (uint32_t)ULONG_MAX;
}    const RawDegrees &rawLat()     { updated = false; return
rawLatData; }    const RawDegrees &rawLng()     { updated = false;
return rawLngData; }    double lat();    double lng();

         TinyGPSLocation() : valid(false), updated(false)
         {}
```

```
                                                                private:
        bool valid, updated;
        RawDegrees rawLatData, rawLngData, rawNewLatData,
    rawNewLngData;       uint32_t lastCommitTime;
void commit();      void setLatitude(const char *term);
       void setLongitude(const char *term);
      };


      struct TinyGPSDate
      {
       friend class TinyGPSPlus;
    public:
       bool isValid() const       { return valid; }
bool isUpdated() const     { return updated; }    uint32_t
age() const       { return valid ? millis() -
      lastCommitTime : (uint32_t)ULONG_MAX; }

        uint32_t value()           { updated = false; return
      date; }
   uint16_t      year();
uint8_t          month();
uint8_t day();

        TinyGPSDate() : valid(false), updated(false), date(0)
        {}
                                                                private:
        bool valid, updated;
      uint32_t date, newDate;
      uint32_t lastCommitTime;    void
      commit();    void setDate(const
      char *term);
      };

      struct TinyGPSTime
      {
       friend class TinyGPSPlus;
    public:
       bool isValid() const       { return valid; }
bool isUpdated() const     { return updated; }    uint32_t
age() const       { return valid ? millis() -
```

```
      lastCommitTime : (uint32_t)ULONG_MAX; }

         uint32_t value()           { updated = false; return
      time; }
        uint8_t hour();
uint8_t minute();
uint8_t second();
uint8_t centisecond();

         TinyGPSTime() : valid(false), updated(false), time(0)
         {}
                                                              private:
        bool valid, updated;
uint32_t time, newTime;         uint32_t
lastCommitTime;           void commit();
        void setTime(const char *term);
      };


      struct TinyGPSDecimal
      {
        friend class TinyGPSPlus;
     public:
         bool isValid() const    { return valid; }    bool
      isUpdated() const  { return updated; }    uint32_t
      age() const    { return valid ? millis() -
      lastCommitTime : (uint32_t)ULONG_MAX; }
         int32_t value()         { updated = false; return val; }

         TinyGPSDecimal() : valid(false), updated(false), val(0)
         {}
                                                              private:
        bool valid, updated;
uint32_t lastCommitTime;
int32_t val, newval;      void
commit();           void set(const
char *term);
      };


      struct TinyGPSInteger
      {
```

```
        friend class TinyGPSPlus;
     public:
        bool isValid() const    { return valid; }       bool
isUpdated() const  { return updated; }    uint32_t age()
const    { return valid ? millis() -
       lastCommitTime : (uint32_t)ULONG_MAX; }
           uint32_t value()        { updated = false; return val; }

         TinyGPSInteger() : valid(false), updated(false), val(0)
         {}
                                                              private:
        bool valid, updated;
uint32_t lastCommitTime;
uint32_t val, newval;           void
commit();           void set(const
char *term);
       };


       struct TinyGPSSpeed : TinyGPSDecimal
       {    double knots()    { return value() / 100.0; }
       double mph()      { return _GPS_MPH_PER_KNOT * value() /
       100.0; }    double mps()      { return _GPS_MPS_PER_KNOT *
       value() / 100.0; }
          double kmph()     { return _GPS_KMPH_PER_KNOT * value()
       / 100.0; }
       };


       struct TinyGPSCourse : public TinyGPSDecimal
       {
          double deg()      { return value() / 100.0; }
       };


       struct TinyGPSAltitude : TinyGPSDecimal
       {
    double meters()       { return value() / 100.0; }    double
miles()       { return _GPS_MILES_PER_METER *  value() / 100.0;
}
          double kilometers()   { return _GPS_KM_PER_METER *
     value() / 100.0; }
          double feet()         { return _GPS_FEET_PER_METER *
value() / 100.0; }
```

```
      };


      struct TinyGPSHDOP : TinyGPSDecimal
      {
         double hdop() { return value() / 100.0; }
      };


     class TinyGPSPlus;
     class TinyGPSCustom
      {
      public:
         TinyGPSCustom() {};
         TinyGPSCustom(TinyGPSPlus &gps, const char
     *sentenceName, int termNumber);    void begin(TinyGPSPlus
&gps, const char *_sentenceName,   int _termNumber);

        bool isUpdated() const  { return updated; }
bool isValid() const    { return valid; }    uint32_t age()
const    { return valid ? millis() - lastCommitTime :
(uint32_t)ULONG_MAX; }
         const char *value()     { updated = false; return
      buffer; }
                                                              private:
         void commit();
         void set(const char *term);


        char stagingBuffer[_GPS_MAX_FIELD_SIZE + 1];
        char buffer[_GPS_MAX_FIELD_SIZE + 1];
unsigned long lastCommitTime;         bool valid,
updated;      const char *sentenceName;           int
termNumber;         friend class TinyGPSPlus;
         TinyGPSCustom *next;
      };


      class TinyGPSPlus
      {
      public:
        TinyGPSPlus();
        bool encode(char c); // process one character received
      from GPS
```

```
  TinyGPSPlus &operator << (char c) {encode(c); return
*this;}

  TinyGPSLocation location;
  TinyGPSDate date;
  TinyGPSTime time;
  TinyGPSSpeed speed;
  TinyGPSCourse course;
  TinyGPSAltitude altitude;
  TinyGPSInteger satellites;
  TinyGPSHDOP hdop;

  static const char *libraryVersion() { return
_GPS_VERSION; }


  static double distanceBetween(double lat1, double
long1, double lat2, double long2);   static double
courseTo(double lat1, double long1, double
lat2, double long2);
  static const char *cardinal(double course);

  static int32_t parseDecimal(const char *term);
static void parseDegrees(const char *term, RawDegrees
&deg);

  uint32_t charsProcessed()   const { return
encodedCharCount; }
  uint32_t sentencesWithFix() const { return
sentencesWithFixCount; }   uint32_t
failedChecksum()   const { return
failedChecksumCount; }
  uint32_t passedChecksum()   const { return
passedChecksumCount; }
                                                  private:
  enum {GPS_SENTENCE_GPGGA, GPS_SENTENCE_GPRMC,
GPS_SENTENCE_OTHER};
```

```
      // parsing state variables
      uint8_t parity;         bool
isChecksumTerm;          char
term[_GPS_MAX_FIELD_SIZE];     uint8_t
curSentenceType;         uint8_t
curTermNumber;     uint8_t
curTermOffset;     bool sentenceHasFix;

      // custom element support
friend class TinyGPSCustom;
TinyGPSCustom *customElts;
   TinyGPSCustom *customCandidates;   void
      insertCustom(TinyGPSCustom *pElt, const char
      *sentenceName, int index);


        // statistics
        uint32_t encodedCharCount;
      uint32_t sentencesWithFixCount;
uint32_t failedChecksumCount;       uint32_t
passedChecksumCount;

   // internal utilities     int
fromHex(char   a);          bool
endOfTermHandler();
      };

      #endif // def(__TinyGPSPlus_h)
```

Lampiran 2

```
/**
* @file       TinyGsmClient.h
* @author     Volodymyr Shymanskyy
* @license    LGPL-3.0
* @copyright  Copyright (c) 2016 Volodymyr Shymanskyy
* @date       Nov 2016
       */

      #ifndef SRC_TINYGSMCLIENT_H_
      #define SRC_TINYGSMCLIENT_H_

     #if defined(TINY_GSM_MODEM_SIM800)
     #include "TinyGsmClientSIM800.h"
 typedef TinyGsmSim800                          TinyGsm;  typedef
TinyGsmSim800::GsmClientSim800            TinyGsmClient;  typedef
TinyGsmSim800::GsmClientSecureSim800
      TinyGsmClientSecure;

      #elif defined(TINY_GSM_MODEM_SIM808) ||
     defined(TINY_GSM_MODEM_SIM868)
      #include "TinyGsmClientSIM808.h"
 typedef TinyGsmSim808                          TinyGsm;  typedef
TinyGsmSim808::GsmClientSim800            TinyGsmClient;  typedef
TinyGsmSim808::GsmClientSecureSim800
      TinyGsmClientSecure;

     #elif defined(TINY_GSM_MODEM_SIM900)      #include
"TinyGsmClientSIM800.h"      typedef TinyGsmSim800
TinyGsm;   typedef TinyGsmSim800::GsmClientSim800
TinyGsmClient;

     #elif defined(TINY_GSM_MODEM_SIM7000)     #include
"TinyGsmClientSIM7000.h"     typedef TinyGsmSim7000
```

```
TinyGsm;   typedef TinyGsmSim7000::GsmClientSim7000
TinyGsmClient;

      #elif defined(TINY_GSM_MODEM_SIM7000SSL)
 #include "TinyGsmClientSIM7000SSL.h" typedef
      TinyGsmSim7000SSL
      TinyGsm;
      typedef TinyGsmSim7000SSL::GsmClientSim7000SSL
      TinyGsmClient;
      typedef TinyGsmSim7000SSL::GsmClientSecureSIM7000SSL
      TinyGsmClientSecure;

      #elif defined(TINY_GSM_MODEM_SIM7070) ||
     defined(TINY_GSM_MODEM_SIM7080) || \
defined(TINY_GSM_MODEM_SIM7090)    #include
"TinyGsmClientSIM7080.h"     typedef TinyGsmSim7080
TinyGsm; typedef TinyGsmSim7080::GsmClientSim7080
      TinyGsmClient;
      typedef TinyGsmSim7080::GsmClientSecureSIM7080
      TinyGsmClientSecure;

      #elif defined(TINY_GSM_MODEM_SIM5320) ||
     defined(TINY_GSM_MODEM_SIM5360) || \
defined(TINY_GSM_MODEM_SIM5300) ||
defined(TINY_GSM_MODEM_SIM7100) #include
"TinyGsmClientSIM5360.h"
  typedef TinyGsmSim5360                    TinyGsm;  typedef
TinyGsmSim5360::GsmClientSim5360 TinyGsmClient;

      #elif defined(TINY_GSM_MODEM_SIM7600) ||
     defined(TINY_GSM_MODEM_SIM7800) || \
defined(TINY_GSM_MODEM_SIM7500)    #include
"TinyGsmClientSIM7600.h"     typedef TinyGsmSim7600
TinyGsm;   typedef TinyGsmSim7600::GsmClientSim7600
TinyGsmClient;

     #elif defined(TINY_GSM_MODEM_UBLOX)       #include
"TinyGsmClientUBLOX.h"       typedef TinyGsmUBLOX
TinyGsm;   typedef TinyGsmUBLOX::GsmClientUBLOX
TinyGsmClient; typedef TinyGsmUBLOX::GsmClientSecureUBLOX
      TinyGsmClientSecure;
```

```
    #elif defined(TINY_GSM_MODEM_SARAR4)     #include
"TinyGsmClientSaraR4.h"     typedef TinyGsmSaraR4
TinyGsm;   typedef TinyGsmSaraR4::GsmClientSaraR4
TinyGsmClient; typedef TinyGsmSaraR4::GsmClientSecureR4
      TinyGsmClientSecure;

    #elif defined(TINY_GSM_MODEM_M95)   #include
"TinyGsmClientM95.h"   typedef TinyGsmM95
TinyGsm;   typedef TinyGsmM95::GsmClientM95
TinyGsmClient;

    #elif defined(TINY_GSM_MODEM_BG96)       #include
"TinyGsmClientBG96.h"       typedef TinyGsmBG96
TinyGsm;   typedef TinyGsmBG96::GsmClientBG96
TinyGsmClient;

      #elif defined(TINY_GSM_MODEM_A6) ||
    defined(TINY_GSM_MODEM_A7)
      #include "TinyGsmClientA6.h" typedef
      TinyGsmA6              TinyGsm; typedef
      TinyGsmA6::GsmClientA6 TinyGsmClient;

    #elif defined(TINY_GSM_MODEM_M590)       #include
"TinyGsmClientM590.h"       typedef TinyGsmM590
TinyGsm;   typedef TinyGsmM590::GsmClientM590
TinyGsmClient;

      #elif defined(TINY_GSM_MODEM_MC60) ||
    defined(TINY_GSM_MODEM_MC60E)      #include
"TinyGsmClientMC60.h"       typedef TinyGsmMC60
TinyGsm;   typedef TinyGsmMC60::GsmClientMC60
TinyGsmClient;

      #elif defined(TINY_GSM_MODEM_ESP8266)
    #define TINY_GSM_MODEM_HAS_WIFI     #include
"TinyGsmClientESP8266.h"    typedef TinyGsmESP8266
TinyGsm; typedef TinyGsmESP8266::GsmClientESP8266
      TinyGsmClient;
      typedef TinyGsmESP8266::GsmClientSecureESP8266
      TinyGsmClientSecure;

      #elif defined(TINY_GSM_MODEM_XBEE)
    #define TINY_GSM_MODEM_HAS_WIFI     #include
"TinyGsmClientXBee.h"       typedef TinyGsmXBee
```

```
TinyGsm;   typedef TinyGsmXBee::GsmClientXBee
TinyGsmClient; typedef TinyGsmXBee::GsmClientSecureXBee
      TinyGsmClientSecure;

      #elif defined(TINY_GSM_MODEM_SEQUANS_MONARCH)
 #include "TinyGsmClientSequansMonarch.h" typedef
      TinyGsmSequansMonarch
      TinyGsm;
      typedef TinyGsmSequansMonarch::GsmClientSequansMonarch
      TinyGsmClient;
      typedef
      TinyGsmSequansMonarch::GsmClientSecureSequansMonarch
          TinyGsmClientSecure;


      #else
      #error "Please define GSM modem model"
      #endif

      #endif  // SRC_TINYGSMCLIENT_H_
```

Lampiran 3

```
#define MODEM_RST            5
#define MODEM_PWKEY          4
#define MODEM_POWER_ON       23
#define MODEM_TX             27
#define MODEM_RX             26
#define LED                  13
#define RELAY                14
#define BLYNK_PRINT Serial
#define TINY_GSM_MODEM_SIM800

#include <TinyGPS++.h>
#include <TinyGsmClient.h>
#include <BlynkSimpleSIM800.h>
// set serial monitor untuk modul gsm dan gps
#define SerialMon Serial
// komunikasi serial untuk modul gsm
#define SerialAT Serial1
// variabel untuk menyimpan data GPS
double latitude;
double longitude;
int satellite;
// use this APN settings if you using Indosat
const char apn[]  = "indosatgprs";
const char user[] = "indosat";
const char pass[] = "indosatgprs";
// auth token
const char auth[] = "tTRE5J-DSIWucdJyRYgqvQhEolfhHbRf";
bool s1;

BlynkTimer timer;

// membuat instance modem untuk modul GSM
TinyGsm modem(SerialAT);

// membuat instance untuk modul GPS
TinyGPSPlus gps;
WidgetMap myMap(V0);
```

```
void GPSData()
 {
   while (Serial.available() > 0)
   {
    if (gps.encode(Serial.read()))
       if (gps.location.isValid())
         {
           unsigned int index = 1;
           latitude = gps.location.lat();
           longitude = gps.location.lng();

           myMap.location(index, latitude, longitude, "Lokasi
           Terkini");
         }
    }
  }


  void GPSCheck()
   {
     if ((gps.charsProcessed() < 10) || (!gps.location.isValid()))
       {
         Serial.println("GPS tidak terdeteksi");
       }
         return;
   }


    BLYNK_WRITE(V3)
    {
      int pinValue = param.asInt();
      if(pinValue==1){

      digitalWrite(RELAY, LOW);
       s1 = true;
       }else{
        digitalWrite(RELAY,HIGH);
       s1 = false;
           }
    }
```

```
void setup() {

 // set baud rate untuk serial monitor
 Serial.begin(9600);
 delay(10);

 // Set-up modem reset, enable, power pins
 pinMode(MODEM_PWKEY, OUTPUT);
 pinMode(MODEM_RST, OUTPUT);
 pinMode(MODEM_POWER_ON, OUTPUT);
 pinMode(LED, OUTPUT);
 pinMode(RELAY, OUTPUT);

 digitalWrite(MODEM_PWKEY, LOW);
 digitalWrite(MODEM_RST, HIGH);
 digitalWrite(MODEM_POWER_ON, HIGH);
 digitalWrite(RELAY, HIGH);

 // pengaturan baudrate untuk modul GSM dan komunikasi UART
 SerialAT.begin(115200, SERIAL_8N1, MODEM_RX, MODEM_TX);
 delay(3000);

 //    ini    digunakan    untuk    restart    modul    GSM
 SerialMon.println("Inisialsasi GSM Modul… ");
 modem.restart();

 String modulInfo = modem.getModemInfo();
 SerialMon.print("Modem: ");
 SerialMon.println(modulInfo);

 SerialMon.print("Menunggu jaringan...");
 if (!modem.waitForNetwork(240000L))
   {
    SerialMon.println(" gagal");
    delay(10000);
   return;
   }
    SerialMon.println(" berhasil");
```

```
if (modem.isNetworkConnected())
   {
     SerialMon.println("Jaringan terhubung");
   }


     SerialMon.print("Menghubungkan ke APN: ");
     SerialMon.print(apn);


 if (!modem.gprsConnect(apn, user, pass))
   {
      SerialMon.println(" koneksi ke GPRS gagal");
      delay(10000);
      return;
   }
      SerialMon.println(" berhasil");


      Blynk.begin(auth, modem, apn, user, pass);


      if (Blynk.connected())
        {
          digitalWrite(LED, HIGH);
          delay(2000);
          digitalWrite(LED, LOW);
        }


 timer.setInterval(2000L, GPSData);
 timer.setInterval(50000L, GPSCheck);

}

void loop()
{
  Blynk.run();
  timer.run();
}
```

LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator

1 Features

- New product available: LMR33630 36-V, 3-A, 400-kHz synchronous converter
- 3.3-V, 5-V, 12-V, and adjustable output versions
- Adjustable version output voltage range: 1.2-V to 37-V ±4% maximum over line and load conditions
- Available in TO-220 and TO-263 packages
- 3-A output load current
- Input voltage range up to 40 V
- Requires only four external components
- Excellent line and load regulation specifications
- 150-kHz Fixed-frequency internal oscillator
- TTL shutdown capability
- Low power standby mode, I_Q, typically 80 µA
- High efficiency
- Uses readily available standard inductors
- Thermal shutdown and current-limit protection
- Create a custom design using the LM2596 with the WEBENCH Power Designer

2 Applications

- Appliances
- Grid infrastructure
- EPOS
- Home theater

3 Description

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version.

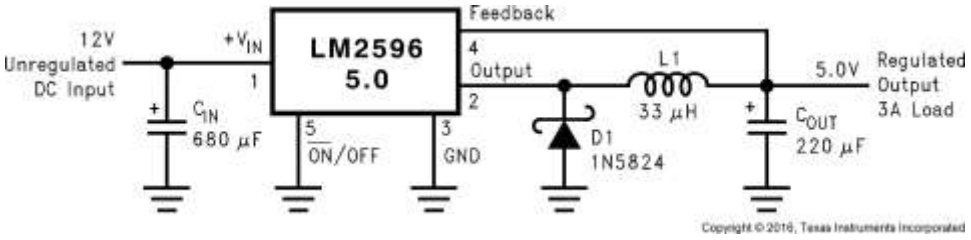
Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation, and a fixed-frequency oscillator.

The LM2596 series operates at a switching frequency of 150 kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators. Available in a standard 5-pin TO-220 package with several different lead bend options, and a 5-pin TO-263 surface mount package.

The new product, LMR33630, offers reduced BOM cost, higher efficiency, and an 85% reduction in solution size among many other features. Start WEBENCH Design with the LMR33630.

PART NUMBER	PACKAGE(1)	BODY SIZE (NOM)
LM2596	TO-220 (5)	14.986 mm × 10.16 mm
	TO-263 (5)	10.10 mm × 8.89 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.



(Fixed Output Voltage Versions)

Typical Application

GPS

locate, communicate, accelerate

NEO-6

u-blox 6 GPS Modules

Data Sheet

Abstract

Technical data sheet describing the cost effective, high-performance u-blox 6 based NEO-6 series of GPS modules, that brings the high performance of the u-blox 6 positioning engine to the miniature NEO form factor.

These receivers combine a high level of integration capability with flexible connectivity options in a small package. This makes them perfectly suited for mass-market end products with strict size and cost requirements.

16.0 x 12.2 x 2.4 mm

www.u-blox.com

2 Pin Definition

2.1 Pin assignment

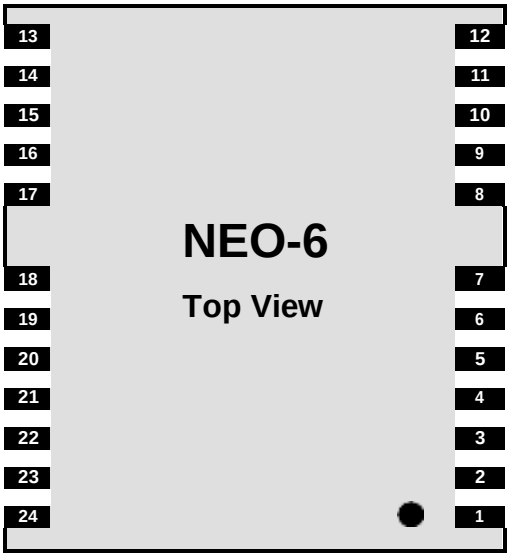


Figure 2 Pin Assignment

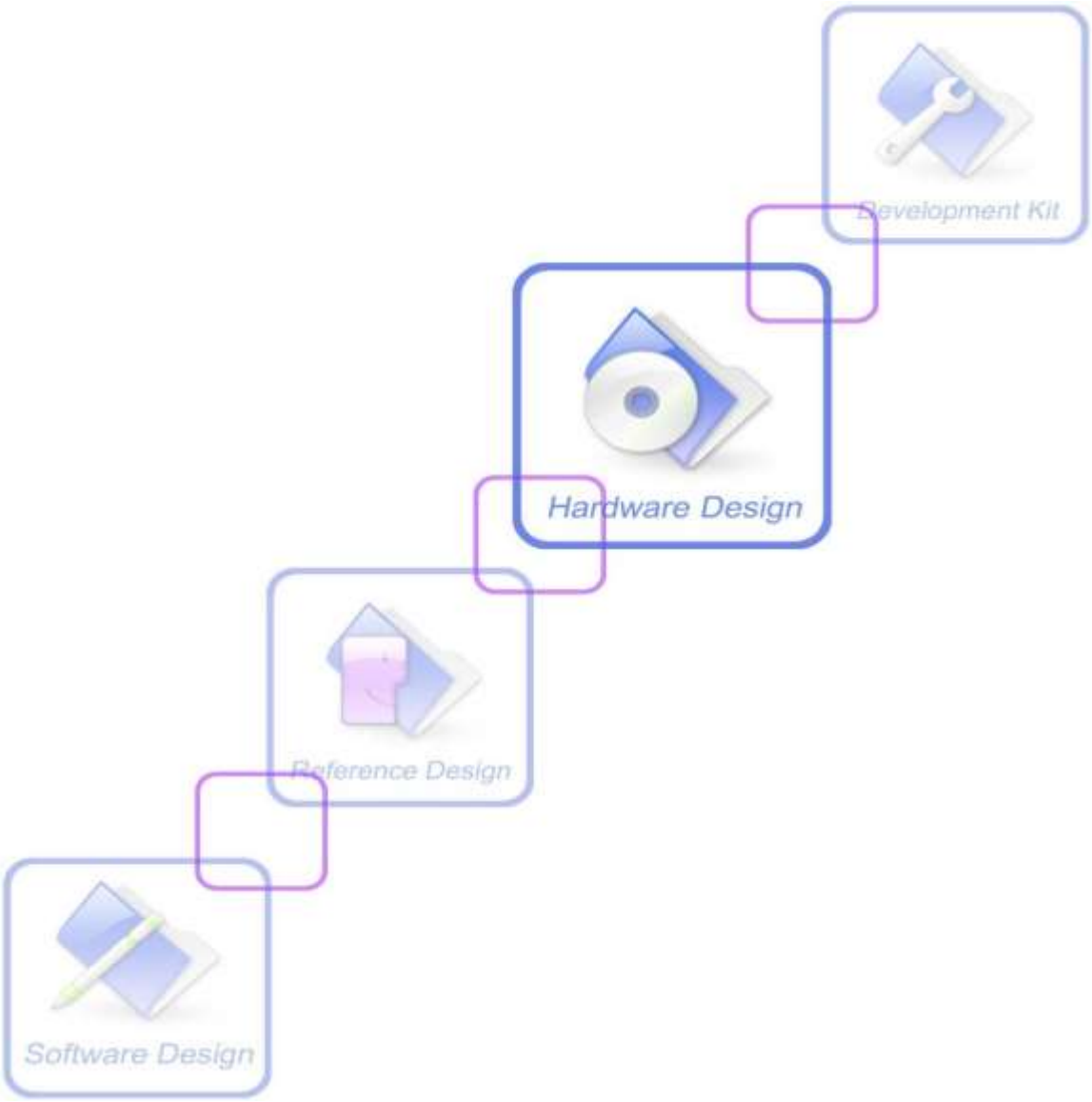
No	Module	Name	I/O	Description
1	All	Reserved	I	Reserved
2	All	SS_N	I	SPI Slave Select
3	All	TIMEPULSE	O	Timepulse (1PPS)
4	All	EXTINT0	I	External Interrupt Pin
5	All	USB_DM	I/O	USB Data
6	All	USB_DP	I/O	USB Data
7	All	VDDUSB	I	USB Supply
8	All	Reserved		See Hardware Integration Manual Pin 8 and 9 must be connected together.
9	All	VCC_RF	O	Output Voltage RF section Pin 8 and 9 must be connected together.
10	All	GND	I	Ground
11	All	RF_IN	I	GPS signal input
12	All	GND	I	Ground
13	All	GND	I	Ground
14	All	MOSI/CFG_COM0	O/I	SPI MOSI / Configuration Pin. Leave open if not used.
15	All	MISO/CFG_COM1	I	SPI MISO / Configuration Pin. Leave open if not used.
16	All	CFG_GPS0/SCK	I	Power Mode Configuration Pin / SPI Clock. Leave open if not used.
17	All	Reserved	I	Reserved
18	All	SDA2	I/O	DDC Data
19	All	SCL2	I/O	DDC Clock
20	All	TxD1	O	Serial Port 1
21	All	RxD1	I	Serial Port 1

No	Module	Name	I/O	Description
22	All	V_BCKP	I	Backup voltage supply
23	All	VCC	I	Supply voltage
24	All	GND	I	Ground

Table 8: Pinout

☞ Pins designated Reserved should not be used. For more information about Pinouts see the *LEA-6/NEO-6/MAX-6 Hardware Integration Manual* [1].

SIM800_Hardware Design_V1.08



3. Package Information

3.1. Pin Out Diagram

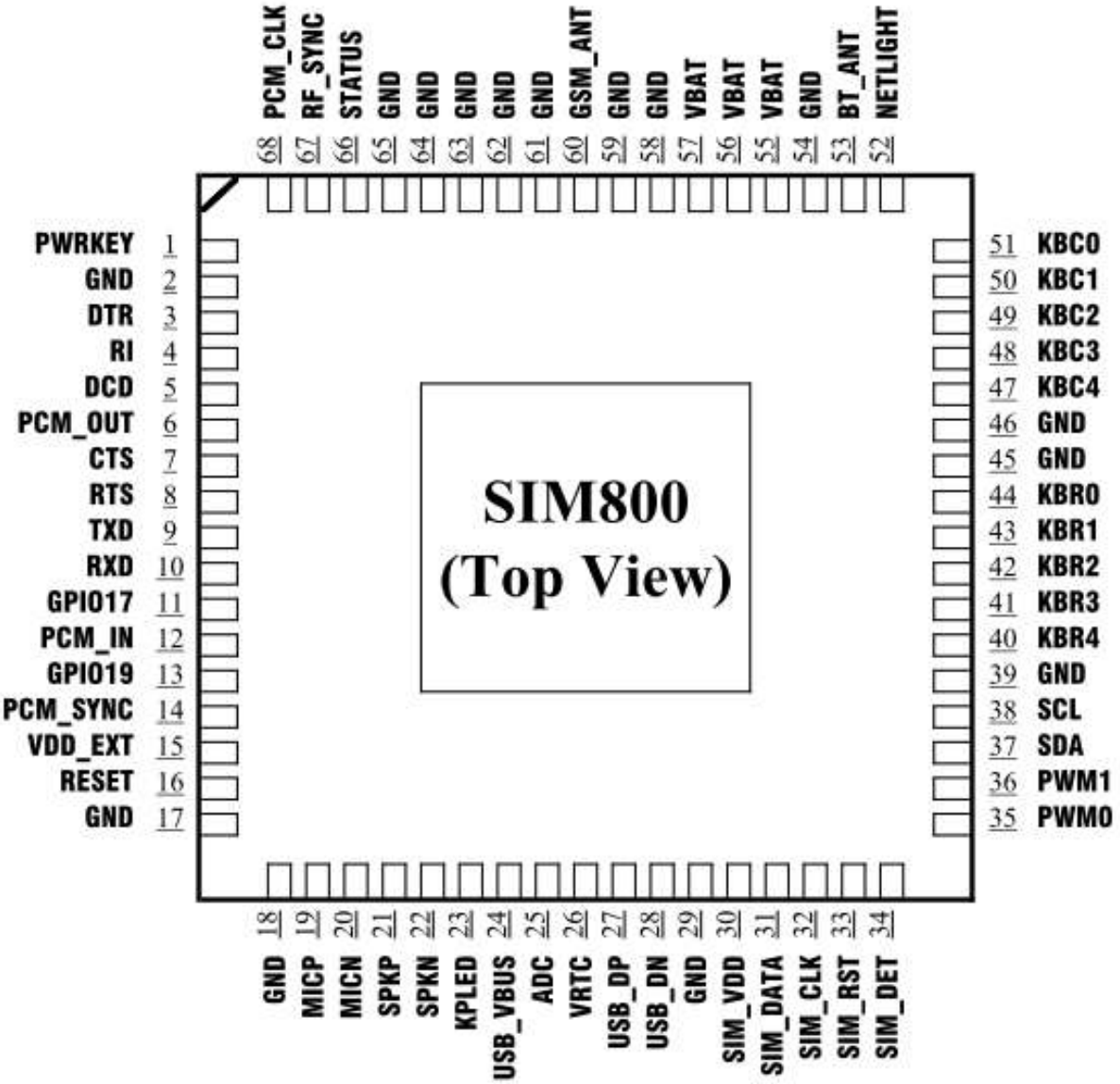


Figure 2: SIM800 pin out diagram (Top view)

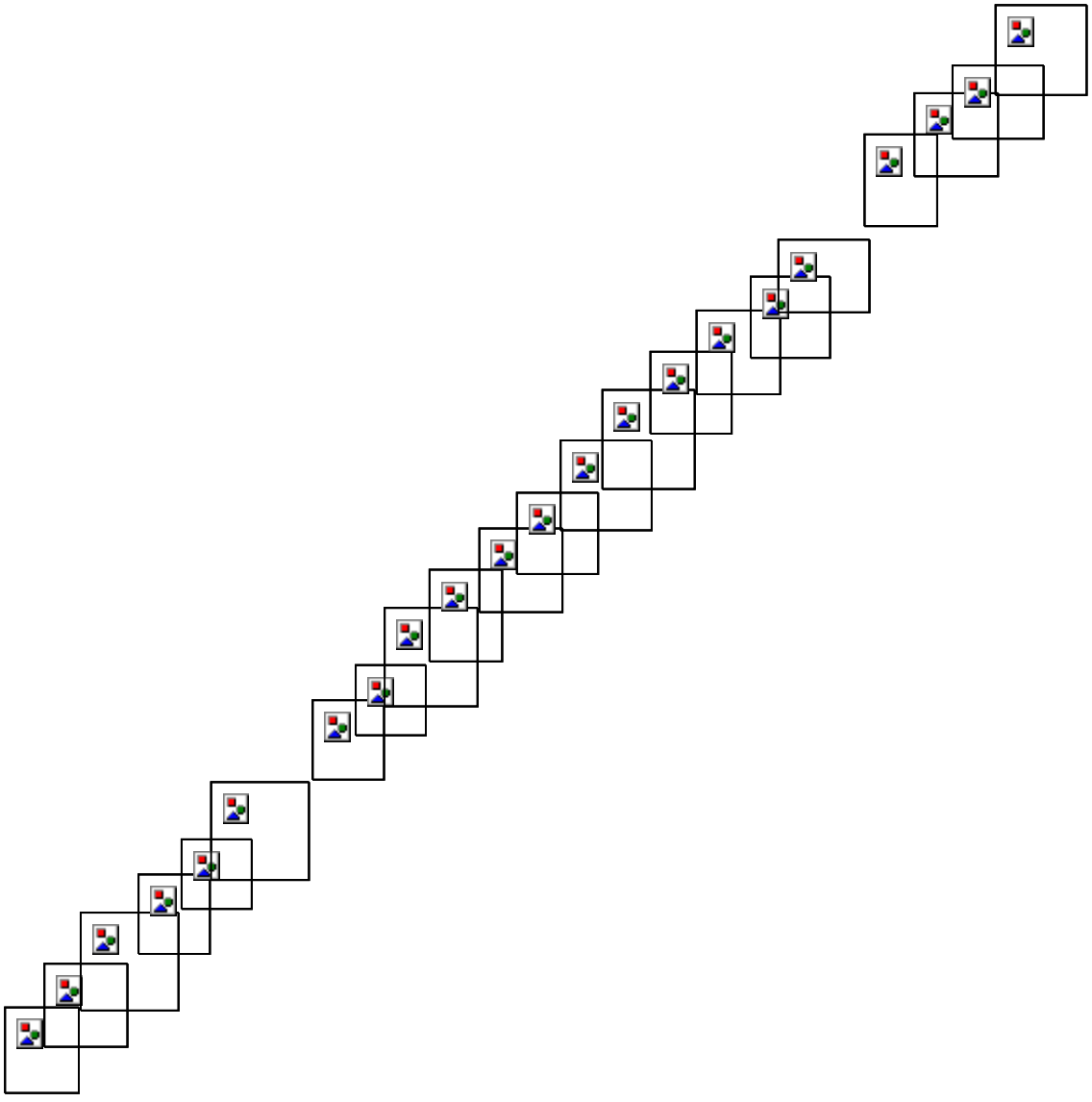
3.2. Pin Description

Table 4: Pin description

Pin name	Pin number	I/O	Description	Comment
Power supply				
VBAT	55,56,57	I	SIM800 supplies 3 VBAT pins, and the power range is from 3.4V to 4.4V. Power supply should provide sufficient current so that the module can work normally; the peak current is nearly 2A.	Zener diode is Strongly recommended to anti surge on VBAT.
VRTC	26	I/O	Power supply for RTC	It is recommended to connect VRTC to a battery or a capacitor (e.g. 4.7uF).
VDD_EXT	15	O	2.8V power output	Keep floating if unused.
GND	2,17,18,29,39,45,46,54,58,59,61,62,63,64,65		Ground	GND for VBAT recommend to use 62, 63, 64, 65 pin .
Power on/off				
PWRKEY	1	I	PWRKEY should be pulled low at least 1 second and then released to power on/down the module.	Internally pulled up to VBAT.
Audio interface				
MICP	19	I	Differential audio input	Keep floating if unused.
MICN	20			
SPKP	21	O	Differential audio output	
SPKN	22			
PCM interface				
PCM_OUT	6	O	PCM interface for audio	Keep floating if unused.
PCM_IN	12	I		
PCM_SYNC	14	O		
PCM_CLK	68	I		
Keypad interface				
KBC4	47	I	Support up to 50 buttons (5*5*2)	Keep floating if unused. (KBC0 can not be pulled down).
KBC3	48	I		
KBC2	49	I		
KBC1	50	I		
KBC0	51	I		
KBR4	40	O		

KBR3	41	O		
KBR2	42	O		
KBR1	43	O		
KBR0	44	O		
GPIO				
GPIO17	11	I/O	Programmable general purpose input and output.	
GPIO19	13	I/O		
NETLIGHT	52	O	Network status	Can not multiplex with GPIO function.
STATUS	66	O	Power on status	
Serial port				
DTR	3	I	Data terminal ready	Keep floating if unused.
RI	4	O	Ring indicator	
DCD	5	O	Data carrier detect	
CTS	7	O	Clear to send	
RTS	8	I	Request to send	
TXD	9	O	Transmit data	
RXD	10	I	Receive data	
USB interface				
USB_VBUS	24	I	Debug and firmware upgrading	Keep floating if unused.
USB_DP	27	I/O		
USB_DN	28	I/O		
ADC				
ADC	25	I	10 bit general analog to digital converter	Keep floating if unused.
PWM				
PWM0	35	O	Pulse-width modulation, multiplex with GPIO22.	Keep floating if unused.
PWM1	36	O	Pulse-width modulation, multiplex with GPIO23.	
I2C				
SDA	37	I/O	I2C serial bus data	Internal pulled up to 2.8V via 4.7KΩ
SCL	38	O	I2C serial bus clock	
SIM interface				
SIM_VDD	30	O	Voltage supply for SIM card. Support 1.8V or 3V for SIM card	All signals of SIM interface should be protected against ESD with a TVS diode array.
SIM_DATA	31	I/O	SIM data input/output	
SIM_CLK	32	O	SIM clock	
SIM_RST	33	O	SIM reset	
SIM_DET	34	I	SIM card detection	
Antenna				
GSM_ANT	60	I/O	GSM antenna port	Impendence must be controlled to 50Ω.

BT_ANT	53	I/O	Bluetooth antenna port	Impendence must be controlled to 50Ω.
RF synchronization				
RF_SYNC	67	O	RF burst synchronous signal	Do not pull up
Other signal				
RESET	16	I	Reset input(Active low)	
KPLED	23	I	Drive keypad backlight	



SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	SRD

1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.
  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

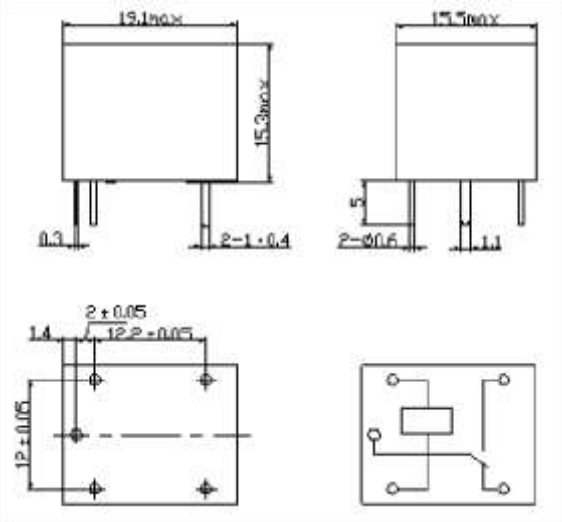
3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03 ▮05 ▮06 ▮09 ▮12 ▮24 ▮48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:CH0052885-2000	7A/240VDC
CCC	FILE NUMBER:CH0036746-99	10A/250VDC
UL /CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R9933789	10A/240VAC 28VDC

5. DIMENSION(unit:mm) DRILLING(unit: mm) WIRING DIAGRAM



6. COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

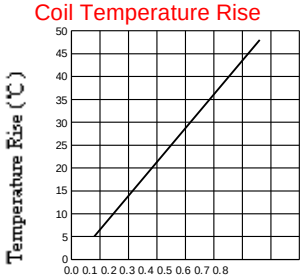
Item \ Type	SRD	
	FORM C	FORM A
Contact Capacity Resistive Load (cosΦ=1)	7A 28VDC 10A 125VAC 7A 240VAC	10A 28VDC 10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

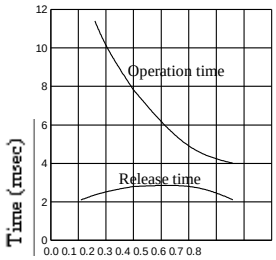
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1minute) 1000VAC 50/60HZ (1minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-25°C to +70°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA

Coil Temperature Rise

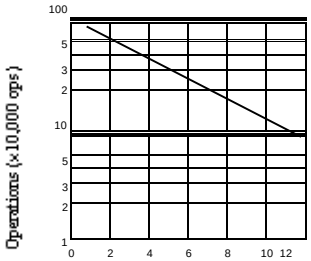


Operation Time



Life Expectancy

AC120V/DC24V cosΦ=1



Life Expectancy